Sistemas UML
Revisão

Prof. Ms. José Antonio Gallo Junior

Introdução à Modelagem de Dados

A modelagem de software é o processo de criar representações abstratas do sistema antes da implementação, garantindo que os requisitos sejam bem compreendidos.

Vantagens da Modelagem:

- Reduz erros e retrabalho.
- Melhora a comunicação entre equipes.
- Facilita a manutenção e escalabilidade do sistema.

Introdução a UML

A **UML** (Unified Modeling Language) é um conjunto de notações gráficas que padronizam a modelagem de software.

Principais Diagramas UML:

- **Estruturais**: Diagrama de Classes, Objetos, Pacotes, Componentes.
- **Comportamentais**: Diagrama de Casos de Uso, Sequência, Atividade.

Processos de Desenvolvimento de Software

São processos definem como o software é projetado, desenvolvido e testado.

Modelo	Características	Vantagens	Desvantagens
Cascata	Sequencial e rígido	Simples e estruturado	Pouca flexibilidade
Iterativo	Desenvolvimento em ciclos	Melhor adaptação a mudanças	Pode ser mais demorado
Ágil	Flexível e colaborativo	Responde rapidamente a mudanças	Difícil gestão
Espiral	Baseado em riscos	Redução de falhas	Custo elevado

Processos e a UML

A UML se adapta a cada modelo:

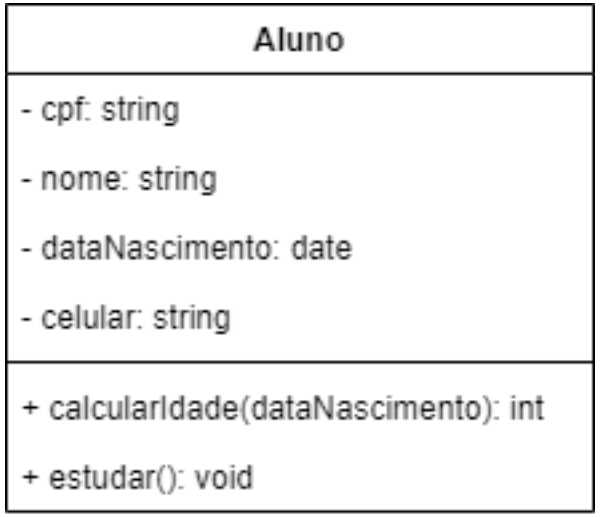
- **Cascata**: Documentação completa antes da implementação.
- **Iterativo**: Modelagem contínua e evolução progressiva.
- **Ágil**: UML simplificada, focada na comunicação.
- **Espiral**: Análise de riscos e validação da arquitetura

Diagrama de Classes

O Diagrama de Classes representa a estrutura estática do sistema. Define classes, atributos, métodos e relacionamentos.

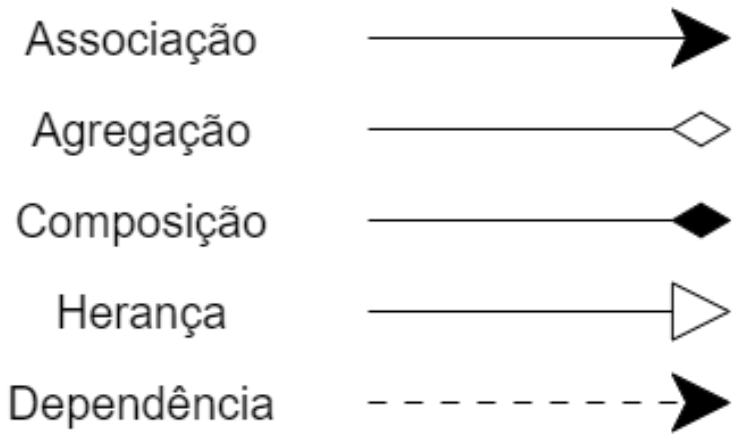
Elementos básicos:

- Nome da classe
- Atributos (dados que descrevem a classe)
- Métodos (ações que a classe pode executar)



Relacionamentos entre Classes

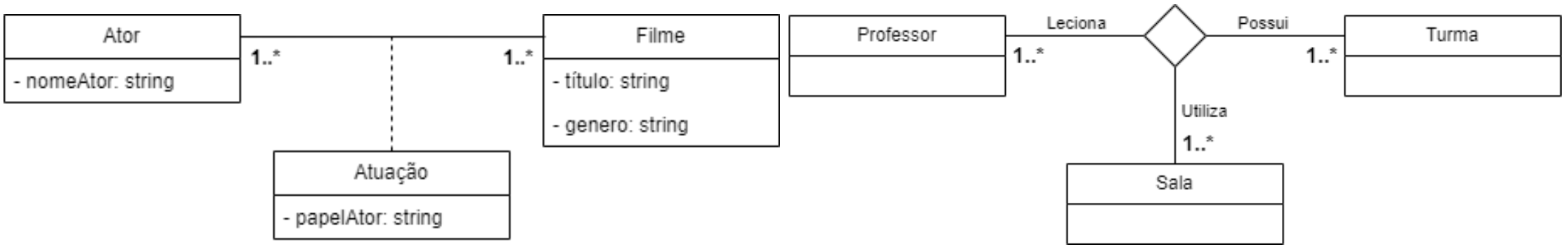
- **Dependência** → Relacionamento fraco, onde uma classe usa a outra temporariamente.
- **Associação**: Ligação entre classes. (Exemplo: Cliente - Pedido).
- **Agregação**: Uma classe pode existir separadamente (Exemplo: Carro e Roda).
- **Composição**: Uma classe depende da outra para existir (Exemplo: Casa e Cômodos).
- **Generalização/Especialização:** Herança entre classes. São usadas para organizar **hierarquias de classes** na UML.



Relacionamentos entre Classes

Ainda temos:

- **Multiplicidade:** Define o **número de objetos** envolvidos na associação.
- **Associação Ternária**: Liga **três classes** ao mesmo tempo.
- **Classe Associativa:** Criada quando há **muitos-para-muitos** com atributos próprios.



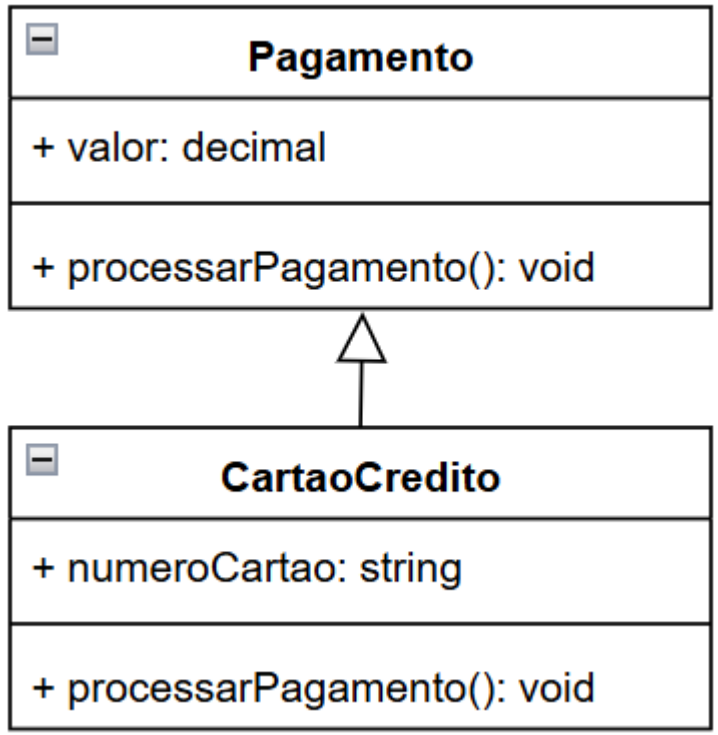
Herança e Polimorfismo

Herança:

- Permite criar novas classes com base em uma classe existente.
- Evita repetição de código e melhora a organização.

Polimorfismo:

- Um método pode ter **comportamentos diferentes** dependendo da classe que o implementa.



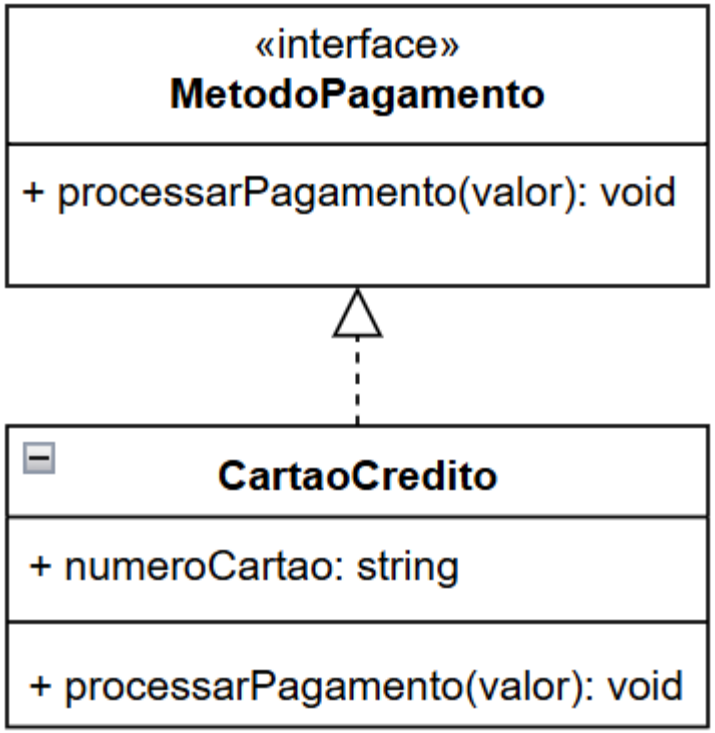
Interfaces e Dependências

Interfaces:

- Definem um contrato de métodos, mas não implementam código.

Dependências:

- Uma classe depende de outra para funcionar.
- Representada por setas tracejadas na UML.



Atividade Prática

Cenário: Sistema de Pedidos Online

Uma empresa deseja um sistema para gerenciar pedidos de clientes em um e-commerce.

Requisitos:

O Cliente pode fazer vários Pedidos.

Cada Pedido pode conter vários Itens de Pedido.

O sistema deve calcular o total do pedido automaticamente.

O Pagamento pode ser realizado via Cartão de Crédito ou Boleto.

Atividade Prática

Tarefa:

- Criar um Diagrama de Classes para representar o sistema.
- Definir atributos e métodos principais.
- Modelar os relacionamentos entre as classes.
- Implementar a herança para diferenciar pagamentos por cartão e boleto.

Atividade Prática